ADDER WITH IMPROVED OVERFLOW FLAG GENERATION

Related Applications

The present application is related to U.S. Patent Application Serial No. 09/291,677 filed April 14, 1999 in the name of inventors M. Besz et al. and entitled "Prefix Tree Adder with Efficient Carry Generation," U.S. Patent Application Serial No. 09/525,644 filed March 15, 2000 in the name of inventors A. Goldovsky et al. and entitled "Prefix Tree Adder with Efficient Sum Generation," and U.S. Patent Application Serial No. 09/569,022 filed May 11, 2000 in the name of inventors A. Goldovsky et al. and entitled "Incorporation of Split-Adder Logic within a Carry-Skip Adder without Additional Propagation Delay," all of which are incorporated by reference herein.

Field of the Invention

The present invention relates generally to adder circuits for use in semiconductor integrated circuits and other electronic devices, and more particularly to techniques for generating overflow flags in such adder circuits.

Background of the Invention

In a conventional n-bit prefix tree adder, the addition of two numbers A and B,

$$A = -a_{n-1}2^{n-1} + \sum_{J=0}^{n-2} a_J 2^J$$

$$B = -b_{n-1}2^{n-1} + \sum_{J=0}^{n-2} b_J 2^J$$

represented in two's complement binary form, can be accomplished by computing:

$$\left.\begin{aligned} g_j &= a_j b_j \\ t_j &= a_j + b_j \\ p_j &= a_j \oplus b_j = \overline{g}_j t_j \\ c_j &= g_j + p_j c_{j-1} \\ s_j &= p_j \oplus c_{j-1} \end{aligned}\right\} \quad \forall j \;\; 0 \le j < n$$

where c_{-1} is the primary carry-input signal. The signals designated g_j, t_j, p_j and c_j are referred to as generate, transmit, propagate and carry signals, respectively. The resulting sum of A and B is

$$S = -s_{n-1}2^{n-1} + \sum_{j=0}^{n-2} s_j 2^j \ .$$

An overflow occurs, and the resulting sum is invalid, if

$$OVF = c_{n-1} \oplus c_{n-2} = 1 \ ,$$

where OVF is an overflow flag.

FIG. 1 shows an example of a logic circuit 100 that may be used to generate the overflow flag OVF in a conventional n-bit prefix tree adder of the type described above. The circuit 100 thus represents a portion of the conventional adder, and in this example includes a two-input OR gate 102, a two-input AND gate 104, a first two-input exclusive-or (XOR) gate 106, and a second two-input XOR gate 108. The primary carry-output signal c_{n-1} is generated as an output of the OR gate 102 and is applied as an input to the XOR gate 108. The carry signal c_{n-2} is applied as an input to both the XOR gate 106 and the XOR gate 108. The XOR gate 106 generates the sum signal s_{n-1} as the exclusive-or of the carry signal c_{n-2} and the propagate signal p_{n-1}. The sum signal s_{n-1} is the final sum bit of the n-bit adder. The XOR gate 108 generates the overflow flag OVF as the exclusive-or

of the carry signals c_{n-1} and c_{n-2}. It is important to note that in this conventional circuit, the overflow flag *OVF* is generated after the addition is completed. Unfortunately, this results in an excessive amount of computational delay.

A gate level model may be used to quantify the computational delay of the FIG. 1 logic circuit 100. In accordance with such a model, a 2-input gate such as a NAND or NOR gate may be specified as having a delay of δ, while XOR/XNOR, AOI (and-or-invert), OAI (or-and-invert) and 2-to-1 multiplexer gates each have a delay of 1.5δ. Using this model, the computational delay associated with generation of the overflow flag *OVF* in the logic circuit 100 is 3δ, corresponding to an adder delay of 1.5δ plus a delay of 1.5δ for the XOR gate 108, as is illustrated in the figure.

Improved adders which can provide significant reductions in logic depth, computational delay and circuit area relative to conventional adders are disclosed in the above-cited U.S. Patent Applications Serial No. 09/291,677, Serial No. 09/525,644 and Serial No. 09/569,022. Although these improved adders provide substantial advantages over conventional adders, a need nonetheless remains for further improvements, particularly in terms of reducing the above-described computational delay associated with overflow flag generation.

Summary of the Invention

The invention provides improved techniques for generation of an overflow flag in a prefix tree adder or other type of adder circuit. In accordance with one aspect of the invention, an overflow flag is generated in parallel with generation of at least one of a sum signal and a primary carry-output signal of the adder. As a result, the computational delay associated with overflow flag generation is substantially reduced. Advantageously, this improvement is provided without requiring any increase in the transistor count or circuit area of the adder.

In an illustrative embodiment of the invention, an n-bit adder includes a number of computational stages each associated with one or more bit positions. Particular ones of the computational stages generate a sum output signal in the form of a final sum bit s_{n-1} of the n-bit adder and a primary carry-output signal c_{n-1} of the n-bit adder. A flag generation circuit is coupled to at least one of the stages and is operative to generate an overflow flag for the adder substantially in parallel with the generation of the sum output signal s_{n-1} and the primary carry-output signal c_{n-1}. The

flag generation circuit is configured such that it does not require the primary carry-output signal c_{n-1} to generate the overflow flag for the n-bit adder.

In accordance with another aspect of the invention, the flag generation circuit for an n-bit adder may generate an overflow flag OVF as:

$$OVF = c_{n-2}\bar{t}_{n-1} + g_{n-1}\bar{c}_{n-2},$$

where c_{n-2} is an n-2 carry signal of the adder, t_{n-1} is an n-1 transmit signal of the adder, and g_{n-1} is an n-1 generate signal of the adder. As a result, the generation of the overflow flag OVF does not require the use of a primary carry-output signal c_{n-1} of the adder.

In accordance with a further aspect of the invention, the flag generation circuit may include a multiplexer which selects one of a plurality of input signals for propagation to its output as the overflow flag based at least in part on a signal associated with the signal line of at least one of the computational stages. For example, the overflow flag OVF described above can be generated using a 2-to-1 multiplexer which receives as a first input a transmit signal t_{n-1}, receives as a second input a generate signal $\bar{g}_{n-1}$, and receives as a select signal a carry signal c_{n-2}. Depending on the value of the select signal, the multiplexer selects either the transmit signal t_{n-1} or the generate signal $\bar{g}_{n-1}$ for propagation in inverted form to its output as the overflow flag.

The efficient overflow flag generation techniques of the present invention provide a substantially reduced computational delay relative to the above-described conventional adder architectures. The techniques of the invention are applicable to a wide variety of adders, including prefix tree adders, carry-lookahead adders, carry-skip adders, carry-ripple adders, carry-save adders as well as other types of adders, and to both radix-2 and non-radix-2 implementations of such adders. These and other features and advantages of the present invention will become more apparent from the accompanying drawings and the following detailed description.

Brief Description of the Drawings

FIG. 1 is a schematic diagram of a conventional logic circuit for computing an overflow flag in an n-bit prefix tree adder.

FIG. 2 is a schematic diagram of a logic circuit in accordance with an illustrative embodiment of the invention for computing an overflow flag in an n-bit prefix tree adder or other type of adder.

FIG. 3 shows an example of an n-bit prefix tree adder in which the logic circuit of FIG. 2 may be implemented.

Detailed Description of the Invention

The present invention will be illustrated below in conjunction with an example prefix tree adder. It should be understood, however, that the invention is not limited to use with any particular type of adder, but is instead more generally applicable to any type of adder in which it is desirable to significantly improve the computational delay associated with generation of an overflow flag without requiring an increase in transistor count or area for the adder circuit, and thus without substantially increasing the cost or complexity of the adder circuit. For example, although illustrated using a radix-2 prefix tree adder, it will be apparent to those skilled in the art that the disclosed techniques are readily applicable to both radix-2 and non-radix-2 adders, as well as to other types of adders, such as carry-lookahead adders, carry-skip adders, carry-ripple adders and carry-save adders.

FIG. 2 shows an example of a logic circuit 200 in accordance with an illustrative embodiment of the present invention. The logic circuit 200 may be used to generate the overflow flag OVF in a conventional n-bit prefix tree adder of the type described above, or in any of a number of other types of adders. The circuit 200 thus represents a portion of such an adder, and in this example includes a two-input OR gate 202, a two-input AND gate 204, a two-input exclusive-or (XOR) gate 206, and a 2-to-1 inverting multiplexer 208. The primary carry-output signal c_{n-1} is generated as an output of the OR gate 202, as in the conventional logic circuit 100 of FIG. 1. In addition, the carry signal c_{n-2} and the propagate signal p_{n-1} are applied as inputs to the XOR gate 206, which generates the sum signal s_{n-1} as the exclusive-or of the carry signal c_{n-2} and the propagate signal p_{n-1}, also as in

the conventional logic circuit 100 of FIG. 1. The sum signal s_{n-1} is the final sum bit of the corresponding n-bit adder.

In accordance with the invention, the overflow flag *OVF* is generated in parallel with generation of the sum signal s_{n-1} and the primary carry-output signal c_{n-1}, using 2-to-1 inverting multiplexer 208 as shown in FIG. 2. More specifically, the following formulation is used to define the overflow flag *OVF* in the illustrative embodiment:

$$OVF = c_{n-1} \oplus c_{n-2} = c_{n-2} \oplus (g_{n-1} + t_{n-1}c_{n-2}),$$

which can be simplified as follows:

$$OVF = c_{n-2}\bar{t}_{n-1} + g_{n-1}\bar{c}_{n-2}.$$

Based on this simplification, the overflow flag *OVF* does not require the use of the primary carry-output signal c_{n-1}, and thus can be generated in parallel with the output sum signal s_{n-1} and the primary carry-output signal c_{n-1}, as is shown in FIG. 2.

The 2-to-1 inverting multiplexer 208 receives as a first input the transmit signal t_{n-1} and as a second input the generate signal $\bar{g}_{n-1}$. The carry signal c_{n-2} is also applied to the multiplexer 208 as a select signal. Depending on the value of the select signal, the multiplexer 208 selects either the transmit signal t_{n-1} or the generate signal $\bar{g}_{n-1}$ for propagation in inverted form to its output as the overflow flag *OVF*. More particularly, if the carry signal c_{n-2} has a logic "one" value, the transmit signal t_{n-1} is propagated in inverted form to the output of the multiplexer 208 as the overflow flag *OVF*, and if the carry signal c_{n-2} has a logic "zero" value, the generate signal $\bar{g}_{n-1}$ is propagated in inverted form to the output of the multiplexer 208 as the overflow flag *OVF*. The multiplexer 208 thus implements the simplified equation given above for the overflow flag *OVF*. Although the

multiplexer 208 in this embodiment is an inverting multiplexer, the invention can be implemented using other types of circuitry.

Using the previously-described gate level model, the computation delay associated with generation of the overflow flag *OVF* in the circuit 200 of FIG. 2 is given by the adder delay 1.5δ, as is indicated in the figure. The additional 1.5δ delay associated with the XOR gate 108 in the conventional logic circuit 100 of FIG. 1 is eliminated in this illustrative embodiment of the invention.

Advantageously, the reduction in the computation time required to generate the overflow flag *OVF* in the illustrative embodiment is achieved without requiring any significant increase in the transistor count or circuit area of the adder, and thus without increasing adder cost or complexity.

It should be emphasized that the logic circuitry in FIG. 2 is shown by way of example only. Those skilled in the art will recognize that numerous alternative arrangements of logic circuitry may be used to reduce the computational delay associated with overflow flag generation in accordance with the techniques of the present invention.

As noted previously, the present invention may be implemented in a number of different types of adders. One such adder will now be described in greater detail in conjunction with FIG. 3.

FIG. 3 shows a set of superimposed prefix trees 300 for an *n*-bit prefix tree adder of the type described in the above-cited U.S. Patent Application Serial No. 09/291,677, and in which the logic circuit 200 of FIG. 2 may be implemented. The general algorithm for an *n*-bit radix-2 prefix tree adder of this type is described below.

Step 1 (1 stage):

Calculate

$$\left.\begin{aligned} g_j &= a_j b_j \\ t_j &= a_j + b_j \\ p_j &= a_j \oplus b_j = \overline{g}_j t_j \end{aligned}\right\} \forall j \;\; 0 \le j < n$$

Step 2 ($\lceil \log_2 n \rceil$ stages):

For $k = 1 \ldots \lceil \log_2 n \rceil$ calculate

$$c_j = G^j_{j-2^{k-1}+1} + T^j_{j-2^{k-1}+1} c_{j-2^{k-1}} \quad \forall j \; 2^{k-1} - 1 \le j < 2^{k-1},$$

$$(G^j_{j-2^k+1}, T^j_{j-2^k+1}) = (G^j_{j-2^{k-1}+1}, T^j_{j-2^{k-1}+1}) o (G^{j-2^{k-1}}_{j-2^k+1}, T^{j-2^{k-1}}_{j-2^k+1}) \quad \forall j \; 2^{k-1} \le j < n.$$

Step 3 (1 stage)

Calculate

$$s_j = p_j \oplus c_{j-1} \quad \forall j \; 0 \le j < n,$$

and

$$c_{n-1} = G^{n-1}_0 + T^{n-1}_0 c_{-1}.$$

In the set of prefix trees of FIG. 3, the squares at the top of the figure compute g_j, t_j and p_j for each bit position in accordance with *Step* 1. The empty circles apply the fundamental carry operator in accordance with *Step* 2. The filled circles represent buffers. The crossed circles compute carries in accordance with *Step* 2 and *Step* 3 above. The diamonds at the bottom of the figure generate the sum at each bit position from the p_j signal in accordance with the equation of *Step* 3. It should be noted that the sum computation of in *Step* 3 occurs in parallel with the computation of the final carry output c_{n-1} in *Step* 3.

In the above description, $(G^j_j, T^j_j) = (g_j, t_j)$, and

$$(G^j_i, T^j_i) = (g_j, t_j) \, o \, (g_{j-1}, t_{j-1}) \, o \ldots o \, (g_i, t_i) \quad \text{if } j > i,$$

where o is the fundamental carry operator. The carry c_j for each bit position is then given by

$$c_j = G_0^j + T_0^j c_{-1}$$

where c_{-1} is the primary carry input. If there is no primary carry input, then c_j is simply G_0^j.

The logic depth of an n-bit prefix tree adder configured as shown in FIG. 3 is $2 + \lceil \log_2 n \rceil$, and the fanout of the carry input c_{-1} is $1 + \lceil \log_2 n \rceil$. The above-described algorithm can also be extended in a straightforward manner to higher radix prefix trees.

Although static circuits are used in the prefix-tree adder 300 of FIG. 3, it should be noted that the invention may be implemented in an adder circuit which includes either static circuits, dynamic circuits or combinations of both static and dynamic circuits. Static circuits are often preferred to dynamic circuits because of their ease of design.

The above-described illustrative embodiment of the invention may be configured to meet the requirements of a variety of different circuit applications, and may be implemented in adder circuits using any desired value of n. Moreover, as previously noted, a variety of other types of adders, including non-radix-2 adders, may also be implemented using the techniques of the present invention.

Adders in accordance with the invention may be used as elements of many different types of circuits, such as, e.g., arithmetic logic units (ALUs), multiply-add units, and comparators. The invention can be incorporated in a wide variety of integrated circuits or other processing devices, including, e.g., microprocessors, digital signal processors (DSPs), microcontrollers, application-specific integrated circuits (ASICs), memory circuits, telecommunications hardware and other types of processing devices.

As a more particular example, the overflow flag generation techniques of the invention can be implemented in a straightforward manner in a DSP such as the DSP16000 from the Microelectronics Group of Lucent Technologies Inc., Allentown, PA, as described in DSP16000 Digital Signal Processor Core, Information Manual, Lucent Technologies, July 1998, which is incorporated by reference herein. In the DSP16000, two modules which can be configured to utilize the overflow flag generation techniques of the invention are the ADDSUB and ALU modules, both described in the above-cited Information Manual.

Numerous alternative embodiments of the present invention within the scope of the following claims will be readily apparent to those skilled in the art.